Analysis of Space Shuttle Main Engine Data Using Beacon-based Exception Analysis for Multi-Missions[1][2]

Han Park, Ryan Mackey, Mark James, Michail Zak
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109
Han.G.Park@jpl.nasa.gov

Michael Kynard, John Sebghati, William Greene
Marshall Space Flight Center
Huntsville, AL 35812

Abstract—This paper describes analysis of the Space Shuttle Main Engine (SSME) sensor data using Beacon-based Exception Analysis for Multimissions (BEAM), a new technology developed for sensor analysis and diagnostics in autonomous space systems by the Jet Propulsion Laboratory (JPL). BEAM anomaly detection system has been applied to SSME in a joint effort between JPL and Marshall Space Flight Center (MSFC). MSFC is evaluating BEAM as an automated tool for rapid analysis of SSME post-ground test data.

BEAM is an end-to-end method of data analysis intended for real-time and non-real-time anomaly detection and characterization. For the SSME application, a custom version of BEAM was built to analyze data gathered during ground tests. Since BEAM consists of modular components, a custom version can be tailored to address specific applications and needs by mixing-and-matching components. The initial build of BEAM focuses on signal processing and contains three components: Coherence-based Fault Detector (CFD), Dynamical Invariant Anomaly Detector (DIAD), and Symbolic Data Model (SDM).

This paper describes the software environment, its training steps, and the analysis results of the SSME data using the DIAD module. DIAD detects anomalies by computing and examining the coefficients of an auto-regressive model, i.e., dynamical invariants. The types of anomalies that can be detected by DIAD are reported such as anomalous sensors, subtle and sudden performance shifts.

TABLE OF CONTENTS

1. INTRODUCTION

The Space Shuttle Main Engine (SSME) is one of the most complex pieces of machinery in operation today. Since it is a safety critical element of the Space Shuttle, it must be tested and certified on a ground test stand before each flight. After each ground test, the results are carefully analyzed for anomalies. Currently, the analysis is conducted primarily by hand, and it relies on the expertise of the data analyst. To reduce time and cost, Marshall Space Flight Center (MSFC) has been evaluating new computational tools that may improve the productivity of the analyst by automating much of the analysis.

Beacon-based Exception Analysis for Multimissions (BEAM) is one of the tools that MSFC has been evaluating. Developed at the Jet Propulsion Laboratory (JPL), BEAM is an end-to-end system of data analysis intended for real-time and off-line anomaly detection and characterization. BEAM was originally developed as an on-board diagnostic system to allow the spacecraft to perform an accurate and timely self-diagnosis. Its compact and modular lends itself naturally to ground-based deployment. Since BEAM consists of modular components, a custom version can be tailored to address specific needs via mixing-and-matching of various components. A custom version of BEAM was built for rapid off-line evaluation of the SSME ground test

[1] 0-7803-7231-X/01/$10.00/© 2002 IEEE
[2] IEEEAC paper #008, Updated Sept 27, 2001

data. The initial architecture of BEAM for the SSME application focuses on signal processing and contains three components: Coherence-based Fault Detector (CFD), Dynamical Invariant Anomaly Detector (DIAD), and Symbolic Data Model (SDM).

This paper describes the software environment, its training steps, and the analysis results of the DIAD component of BEAM. DIAD detects anomalies by computing and examining the coefficients of an auto-regressive model. The results for some of the more common types of anomalies encountered during SSME testing, such as anomalous sensors, subtle and sudden performance shifts, are presented.

2. BEAM OVERVIEW

BEAM is a complete data analysis system for real-time or off-line fault detection and characterization. While the originally intended for generic system analysis on-board deep space probes and other highly automated systems, the compact and modular nature of its subroutines naturally lends itself to ground-based deployment. For the SSME application, BEAM was used as an automated graphical tool for analysis of the post-ground test data.

The basic premise of BEAM is to construct a strategy to characterize a system from all available observations, and train such characterization with respect to normal phases of operation. In this regard, the BEAM engine functions much as a human operator. Through experience and other available resources (known architecture, models, simulation, etc.) an allowed set of behavior is "learned" and deviations from this are noted and examined. Such sophisticated approach should be applied as a complement to less complex traditional monitors and alarms found in nearly all instrumented systems. The approach should also be constructed such that information products can be used to drive autonomous decisions or to support the decisions of human operators. In other words, the system must alert and provide support for intervention wherever possible. Otherwise, it will be difficult for spacecraft experts to gain trust in the system, and the benefits of BEAM or any similar cost-saving approach will be doomed from the outset. In this manner BEAM is not only suited to beacon monitoring but also more broadly applicable to monitored or wholly autonomous systems.

At the simplest level of abstraction, BEAM is software, which takes data as input and reports fault status as output. Implementation of this software dependents on the application, but a typical application would have a system with a number of individual components, each of which reports health or performance data to a local computer. To accommodate such a wide range of possibilities, the computational engine of BEAM itself is highly adaptable with respect to subsystem size and complexity.

For each single compartment or subsystem, we can expect to receive four types of data:

1. Discrete status variables changing in time – modes, switch positions, health bits, etc. – from sensors or software
2. Real-valued sensor data varying at fixed rates – performance sensors or dedicated diagnostic sensors
3. Command information – typically discrete as in 1.
4. Fixed parameters – varying only when commanded to change but containing important state information

These types of data are all valuable but used in different ways. Status variables and commands are useful to a symbolic model. Commands and fixed parameters are used in a physical system model while the time-varying sensor data are used in signal processing components. An optimal strategy must take each of these into account and produce a single, unified decision. In order to study each and combine results, we propose the following BEAM architecture. (Figure 1)

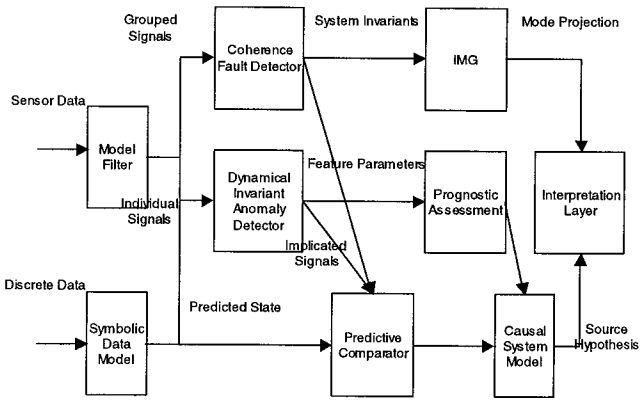

Figure 1. Top-level BEAM architecture.

A simple description of each module is given as follows. A more complete description can be found in [1].

1. *Model Filter (MF):* Receives sensor or other quantitative data, conditions it in terms of synchronicity or drop-outs, and combines results with physics model (simulation) predictions.
2. *Coherence Detector (CD):* Receives multiple conditioned quantitative signals and performs anomaly detection using cross-signal statistical features.
3. *Dynamical Invariant Anomaly Detector (DIAD):* Receives a single quantitative signal one at a time and performs anomaly detection using a parametric estimate of the residuals.
4. *Informed Maintenance Grid (IMG):* Combines outputs

from the Coherence Detector over long operating periods to detect subthreshold degradation and predict functional failures.

5. *Prognostic Assessment (PA):* Uses the parametric signal estimates from DIAD to forecast future signal values and identify potential signal faults.
6. *Symbolic Data Model (SDM):* Receives discrete data and constructs an internal state estimate of the system. It detects discrete signal mismatches (explicit faults) and identifies system mode for use by other components.
7. *Predictive Comparator (PC):* Combines signal implications from the CD and DIAD as well as discrete reports from SDM in an attempt to unify results from the signal-based and symbolic reasoning components.
8. *Causal System Model (CSM):* Backtracks implications along the system structure to reduce the complexity of fault reports. This is a simplistic form of diagnosis.
9. *Interpretation Layer (IL):* Fuses results from different components and constructs a final report. It serves as an interface between BEAM and other components.

3. DYNAMICAL INVARIANT ANOMALY DETECTOR

The Dynamical Invariant Anomaly Detector is a component of BEAM designed to identify and isolate anomalies in the behavior of individual sensor data. The full mathematical details of the DIAD can be found in [2]. Traditional methods detect abnormal behavior by analyzing the difference between the sensor data and the predicted value. If the values of the sensor data are deemed either too high or low, the behavior is abnormal. In our method, we introduce the concept of *dynamical invariants* for detecting structural abnormalities.

Dynamical invariants are governing parameters of the dynamics of the system, such as the coefficients of the differential (or time-delay) equation in the case of time-series data. Instead of detecting deviations in the sensor data, which can occur simply due to different initial conditions or external forces, i.e. operational anomalies, we attempt to identify structural changes or behavioral changes in the system dynamics. While an operational abnormality will not lead to a change in the dynamical invariants, a true structural abnormality will lead to a change in the dynamical invariants. In other words, the detector will be sensitive to problems internal to the system and not external disturbances.

We start with a description of a traditional treatment of sensor data given in the form of a time series describing the evolution of an underlying dynamical system. The method assumes that the dynamics are captured in the sensor data. It will be assumed that this time series cannot be approximated by a simple analytical expression and does not possess any periodicity. In other words, for an observer, the future values of the time series are not fully correlated with the past ones, and therefore, they are apprehended as random. Such time series can be considered as a realization of an underlying stochastic process, which can be described only in terms of probability distributions. However, any information about this distribution cannot be obtained from a simple realization of a stochastic process unless this process is stationary - in which case, the ensemble average can be replaced by the time average. An assumption about the stationarity of the underlying stochastic process would exclude important components of the dynamical process such as linear and polynomial trends, or harmonic oscillations from consideration. Thus we develop methods to deal with non-stationary processes.

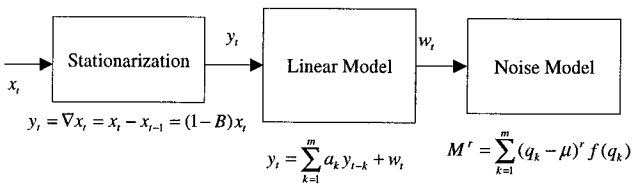

Figure 2. Data flow diagram for Dynamical Invariant Anomaly Detector.

The data flow diagram is shown in Figure 2. First, the sensor data are stationarized using the difference operator [3]. Then the stationarized data is fed into a memory buffer, which keeps a time history of the sensor data for analysis. We study the critical signals, as determined by the symbolic components of BEAM, the operating mode. The relevant sensor data is passed to the autoregressive parameter estimator. The autoregressive process is given by:

$$x(t) = a_1 x(t-1) + a_2 x(t-2) + \ldots + a_n (t-n) + z(t), \quad (1)$$

where a_i are the parameters, i.e., dynamical invariants, and $z(t)$ represents the contribution from white noise. The dynamical invariants, a_i, are computed using the Yule-Walker equations.

Once the parameters are computed, they are compared to the ones stored in a model parameter database. This contains a set of nominal time-delay equation coefficients appropriate for particular operating mode. A statistical comparison is made between the stored and just-computed coefficients, and if a discrepancy is detected, i.e., a parameter value exceeds the nominal confidence limit, the sensor data is identified as being anomalous.

Further analysis can be carried out on the residual or the difference between the sensor data values and the model predicted values, i.e. the uncorrelated noise, using a nonlinear neural classifier and/or noise analysis techniques.

The nonlinear neural classifier is designed to extract the nonlinear components, which may be missed by the linear Yule-Walker parameter estimator. The weights of the artificial neural network, another set of dynamical invariants, will be computed and compared with nominal weights stored in the model parameter database. Similarly, the noise characteristics, such as the moments of probability distribution, are dynamic invariants for the stationarized sensor data, and can be compared with those stored in the Model Parameter Database. If any anomalies are detected in either the nonlinear components or the noise, the identity of the sensor will be sent to the channel anomaly detector.

4. Space Shuttle Main Engine

The cluster of three SSMEs mounted on the aft of the Space Shuttle orbiter is an integral and primary element of the Space Shuttle propulsion system. The SSME is a high energy density reusable rocket engine employing cryogenic liquid hydrogen and liquid oxygen as propellants. It is capable of a broad range of throttle settings, allowing it to meet the requirements for dynamic load mitigation during launch ascent, and simultaneously control engine thrust level and engine mixture ratio via closed-loop control systems. The SSME of today is the product of over a quarter century of development evolution, hundreds of thousands of seconds of accumulated ground test time, and well over one hundred successful Space Shuttle launches.

The SSME consists of a dual-preburner, staged combustion cycle driven by a combination of propellant boost pumps and high-pressure turbopumps. The structural backbone of the SSME, the powerhead, supports the three combustion chambers within the engine and provides mounting location for the two high-pressure turbopumps. Also part of the engine package are the two low-pressure turbopumps, five hydraulic/pneumatic control valves and actuators, high and low connecting pressure ducts, and the regeneratively cooled nozzle extension.

An on-board controller is mounted to each engine. This unit acts as the active, closed-loop control device and is the collection site for the engine instrumentation data during an engine firing. SSME instrumentation, approximately 90 measurements including pressures, temperatures, rotational speeds, and accelerometer data, falls within three general categories: those used for engine thrust and mixture ratio control, those used to protect engine structural limits (redlines), and those used for data analysis and hardware maintenance.

The operation of the SSME can be viewed as consisting of four phases. The first phase of the operation is the pre-start period during which the engine is purged and conditioned with the cryogenic propellants. This phase is crucial for a reliable and consistent engine start. The second phase of the operation is the start phase. This period of approximately six seconds spent on the launch pad as the thrust builds to nominal, steady state operation provides for an opportunity to ensure the proper operation of the engine prior to leaving the ground. The third phase of the operation, called mainstage, is where the SSME provides thrust to the ascent of the Space Shuttle and lasts nearly nine minutes. And finally, the fourth phase of the operation is engine shutdown, which is a controlled exercise in making the safe transition from powered ascent to orbital operations.

The components that make up each SSME are demonstrated to be worthy of launch operations through full engine system testing conducted at the NASA Stennis Space Center in Mississippi. Here, on test stands customized to simulate launch conditions, tests of flight hardware are conducted along with other tests intended to ensure proper engineering characterization of engine component reliability.

5. Training

For the SSME data analysis, a custom MATLAB application with a graphical user interface was developed for DIAD training and evaluation as shown in Figure 3. The application is menu driven and features a complete set of tools necessary for stationarizing and modeling the sensor data.

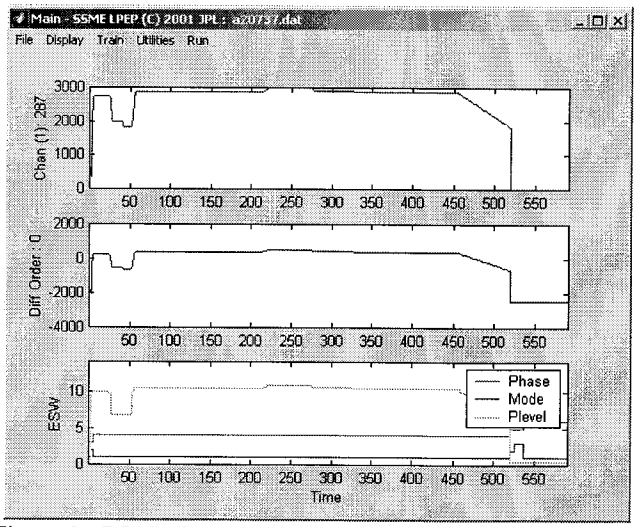

Figure 3. DIAD GUI application.

The training of DIAD is entirely data-driven. DIAD is trained with examples of nominal operating data covering the major phases and modes of operation for each channel. For the SSME application, the DIAD was trained using sixteen nominal ground test data from Block II engine series. The tests covered various power levels (100, 104.5, 106, 109, 111% thrust levels) as well as several different engines. Each test data contained 100 channels of sensor data covering the major subsystems on the SSME. DIAD monitored 86 critical engine sensors out of the 100 sensor data. Fourteen channels were control parameters such as

engine commands to identify the mode of operation. The two critical control parameters were the Engine Status Word and Commanded Main Combustion Chamber Pressure. They were used to deduce and identify the phase, mode, and power level of the engine.

The training steps for DIAD were identical for each channel of the 86 channels. They are as follows:

1. Segment the data by the phase, mode, and power level. For main stage phase, five power level bins, centered at 100, 104.5, 106, 109, and 111%, were used. The bin centers were located at the power level settings normally encountered during ground testing as shown in Figure 4. A temperature sensor data has been segmented for main stage phase, normal control mode, and 109% power level.

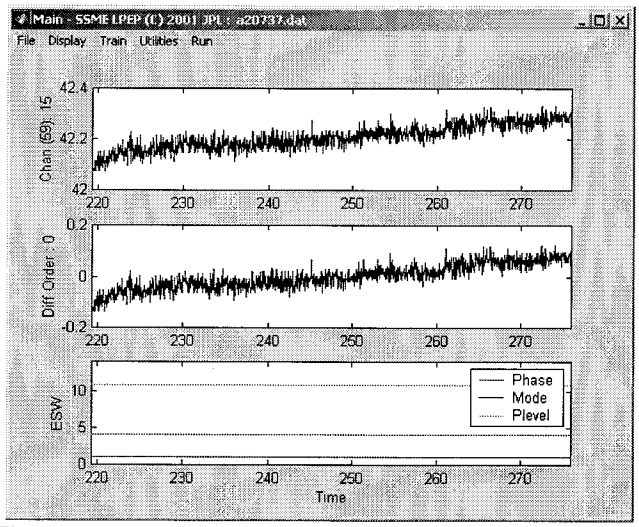

Figure 4. Segmented data. The data has been segmented for main stage phase, normal control mode, and 109% power level.

2. Compute the best difference order to stationarize the data. The order of differencing can be estimated using the Partial Auto-Correlation (PAC) function as shown in the left column plots in Figure 5. Refer to [3] for more details on PAC. The best differencing order is usually the one that minimizes the sum of the PAC as shown in the upper right plot in Figure 5. In this case, the first order differencing was the best order. The bottom right plot is the Power Spectral Density (PSD) plot of the data. If there are any strong peaks in the PSD, seasonal differencing, i.e., periodic differencing, is required. The current version GUI version of DIAD does not support seasonal differencing so the plot is currently for diagnostics only. Fortunately, the sensor data in the case of SSME did not show any strong periodic components.

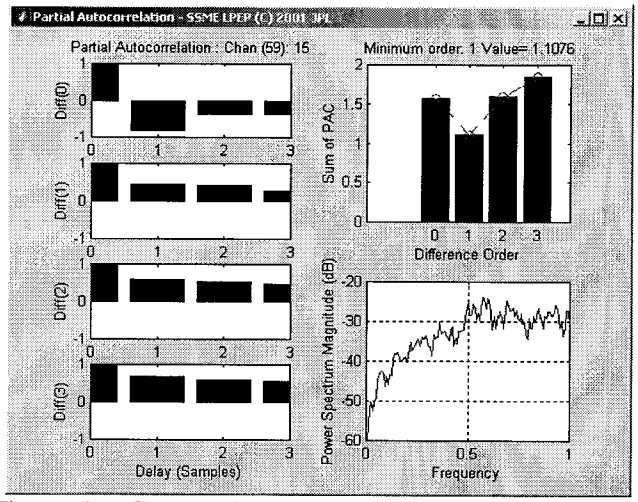

Figure 5. Computation of the best difference order to stationarize the data.

3. Estimate the order of the Autoregressive (AR) model. The order of the AR model can be estimated again using the PAC. For a given sampling window size, it is possible to estimate the order of the model by simply computing the standard error limits of the PAC [3]. In the example above, the estimated order is two using a sampling window size of 100 samples. The model fit is shown in the middle plot of Figure 6. The green line is the estimated data using the second order AR model. The bottom plot shows the residual or the noise component. This component is characterized by its moments.

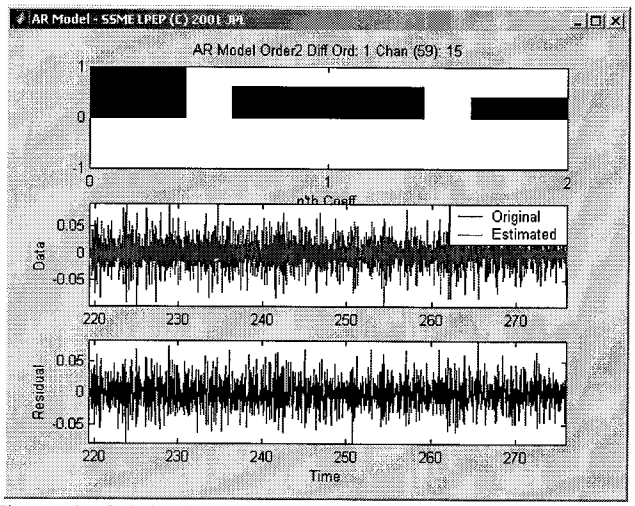

Figure 6. Original versus estimated data using AR model.

4. Compute the mean and confidence level of the AR parameters. As shown in Figure 7, the mean and the 99% confidence level of the AR parameters are shown by the solid blue and dashed red lines, respectively. These values are computed by moving a window of size 100 samples by increments of 10 samples. These are the nominal values of the parameters. If the parameters exceed the 99% confidence level, the channel will be tagged as anomalous. For ease of computation, the sum of the AR parameters

(shown in the bottom plot) is used for monitoring the data for anomalies.

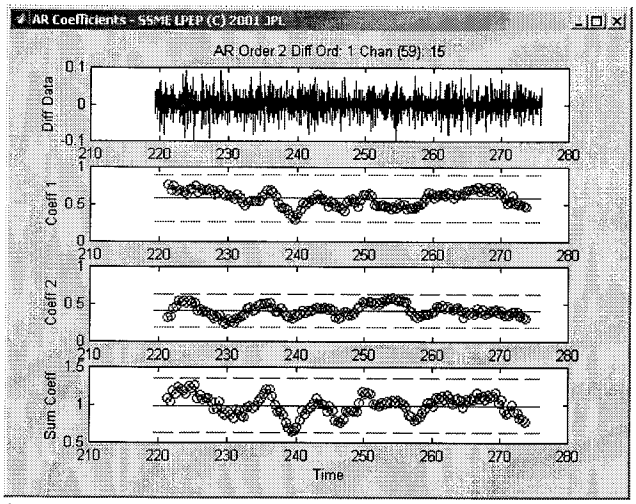

Figure 7. Mean and the 99% confidence level for the AR parameters.

These steps complete the training of the DIAD. As noted above, they are carried out for each channel. Since there are 86 channels, this process can be quite laborious. Therefore, an auto-training feature was added which automated the process. That is the user simply selects the phase, mode, and power level, and the rest is done automatically.

There are some caveats to training in terms of the data sets. First, insufficient data training will result in false alarms, indicating novelty, until data collection and review during or before flight operations produce a sufficiently large data set to cover all nominal operating modes. Second, the method is only likely to be effective if the system dynamics are captured in the sensor data. If the sensor has too much noise or is purely random, the DIAD may have difficulty extracting any useful information out of the signal. Finally, if there is too much variation between different systems, e.g. variation between different engines of an identical model, DIAD can become somewhat insensitive. This issue is revisited in the next section.

6. RESULTS

After training DIAD using sixteen nominal test data from Block II engine series, its anomaly detection capabilities were evaluated on seven different anomalous data sets. These test data contained some of the most commonly encountered anomalies in SSME testing. These include High Pressure Fuel Turbo Pump (HPFTP) blade failure, Low Pressure Fuel Turbo Pump (LPFTP) and HPFTP cavitation, High Pressure Oxidizer Turbine (HPOT) performance shift, fuel flowmeter shift, frozen sense lines, and deactivated sensors.

For anomaly detection, we first compute the difference sum of the current and nominal AR parameters (stored in the Model Parameter Database), ζ, as given by:

$$\zeta = \sum \left[\left(a_i - \overset{o}{a}_i \right)^2 \right]. \qquad (2)$$

where $\overset{o}{a}_i$ are the nominal values of the parameters, and a_i, are their current values. If ζ exceeds $|\varepsilon|$ where ε is the 99% confidence limit of the parameters, the system is deemed anomalous. The advantage of this criterion is in its simplicity. It can be periodically updated, and therefore, the health of the system can be easily monitored.

The first example of anomaly detection is a frozen sense line. When a sensor freezes due to its cryogenic environment, its dynamics can change as shown in Figure 8. At time = 180 sec, the sense line freezes causing the sensor value to drop and have different dynamics.

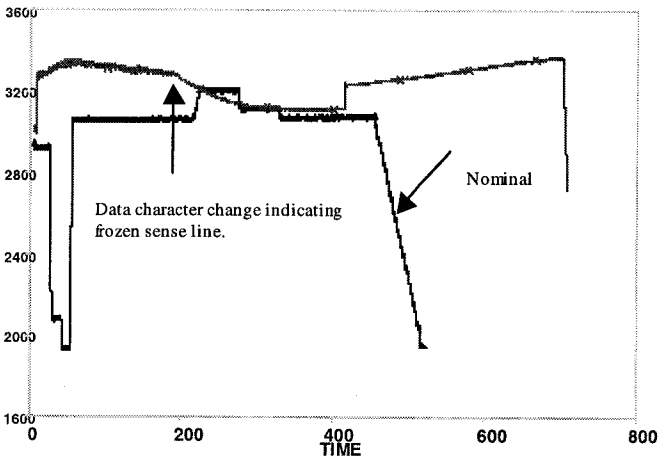

Figure 8. Comparison between nominal versus frozen sense line.

The normalized difference sum, ζ, is shown in the middle plot of Figure 9. If ζ exceeds 1, the sensor dynamics are anomalous. The plot shows a jump in ζ after time = 180 sec. The value of ζ persistently stays above the 99% confidence limit throughout the remainder of the test. At time = 415 sec, there is a large spike in ζ due to the sudden jump in the sensor value as clearly shown in Figure 8. The jump was caused when the sense line became partially unfrozen and the sensor value jumped. However, DIAD shows that the sensor continues to have anomalous dynamics even after this event.

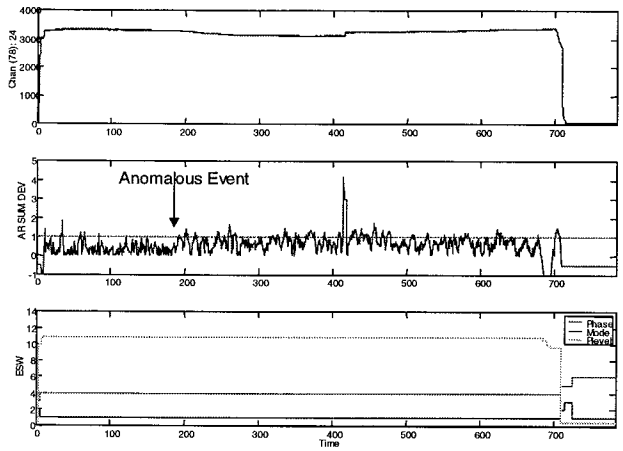

Figure 9. Normalized difference sum of the current and nominal AR parameters. The difference sum is shown in the middle plot. Note the jump at t = 180 sec. The top plot is the raw data while the bottom plot is the phase, mode, and power level of the engine.

Figure 10 shows the overall system view of the sensors. The vertical axis represents the sensor channels while the horizontal axis is time. The colors indicate the degree of deviation from nominal parameters. Dark blue indicates regions where the DIAD was not trained to make proper determination of sensor dynamics. The first 14 channels are control parameters, so they were ignored and thus marked dark blue. The other dark areas are regions where the DIAD was not properly trained for the particular phase, mode, and power level. As discussed earlier, the DIAD was trained only for main stage phase, and normal control mode in this evaluation. The startup ($0 < t < 10$ sec) and shutdown ($t > 720$ sec) regions are therefore marked dark blue.

The interesting anomalies are shown in red where the difference sum exceeds the confidence limit. There are three regions where red colors are observed. The red streaks around channel 30 caused by bit-toggling data are due to low analog-to-digital converter resolution, which indicate several false alarms. These channels may need to be modeled using a difference process, perhaps a Hidden Markov Model.

The bright red spots at time t = 450 sec at channels 81, 84, and 85 were caused by Gaseous Oxygen (GOX) repressurization of the oxygen tank. This event was planned and caused a momentary jump in the sensor data.

Finally, the persistent deviation in channel 78 is caused the frozen sense line as shown in Figure 8. The anomaly begins around t = 180 sec and persists throughout the test.

As demonstrated, the system view plot of anomalies gives an analyst a bird's eye view of the sensor dynamics. If any sudden red color is detected, the analyst can quickly bring up the raw sensor and individual DIAD results plots for further analysis.

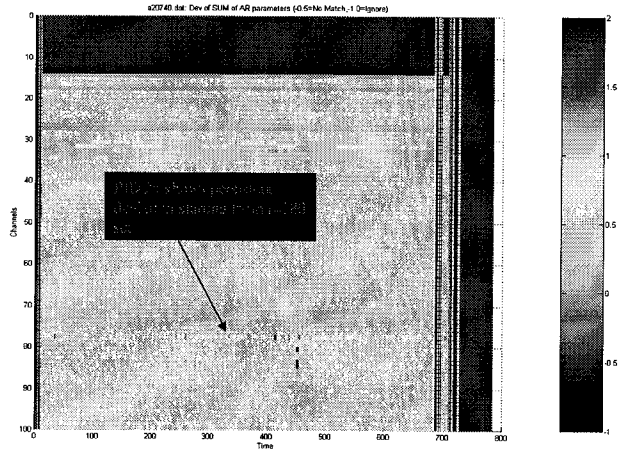

Figure 10. Overall system view plot. The frozen sense line anomaly is at channel 78 as indicated by the arrow.

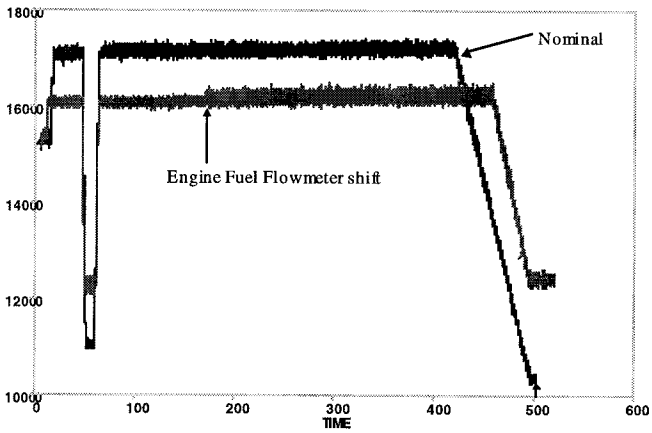

Figure 11. Comparison between a nominal and shifted fuel flow meter.

Another type of common anomaly is shown in Figure 11, in which the engine fuel flow meter has shifted. The shift is believed to be caused by vortex shedding off the upstream flow straighteners. There appear to be operational regimes, very roughly analogous to resonance points, where the vortex-to-flowmeter blade interactions cause a bistability. Unfortunately, the precise operational conditions causing this effect cannot be predicted with first order data analysis and modeling them varies dramatically from unit to unit. Examining the normalized difference sum, ζ, as shown in the middle plot of Figure 12, reveals that there is a jump in ζ at t = 180 sec, corresponding to the fuel flow meter shift at the same time. It is noteworthy that the normalized ζ did not quite exceed the 99% confidence limit. A similar situation was observed for a frozen sense line anomaly as shown in Figure 13. It shows a jump in the AR parameters at t = 275 sec corresponding to the frozen sense line.

However, the parameters did not exceed the 99% confidence limit.

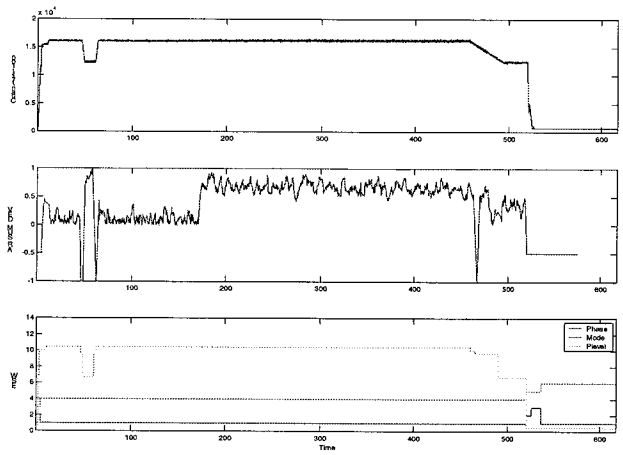

Figure 12. Normalized difference sum of the current and nominal AR parameters. The difference sum is shown in the middle plot. Note the jump at t = 180 sec. The top plot is the raw data while the bottom plot is the phase, mode, and power level of the engine.

The cause can be traced back to the large variations in the training data used in the training procedure. In certain cases, the variability in some sensor data between different engine tests (training files) was so large that the 99% confidence limit was set very wide during training, resulting in an insensitive detector. While the majority of the sensor data were consistent across various engines, certain sensor characteristics were unique to each engine. Unfortunately, these variations from engine to engine may be unavoidable since each engine is uniquely assembled.

The solution to this problem may be addressed in one of two ways. First is to have unique training set for each engine. The drawback of this solution is that a unique set of nominal parameters is required for each engine, which may lead to logistical and bookkeeping difficulties. In addition, anomaly detection is not possible on a new engine until it is fully tested at each operating condition.

The solution we are currently pursuing is to look for clusters of nominal parameter values. It has been observed that some sensor data dynamics exhibit clustering behavior. In other words, certain groups of engines exhibit similar sensor behavior. This is reasonable since Block II engines are built in small production runs. Within each production run, the engines may be very close to each other. The solution is to build a set of selected parameters that best match the engine production run. This will require a clever clustering algorithm which we are in the process of developing and evaluating.

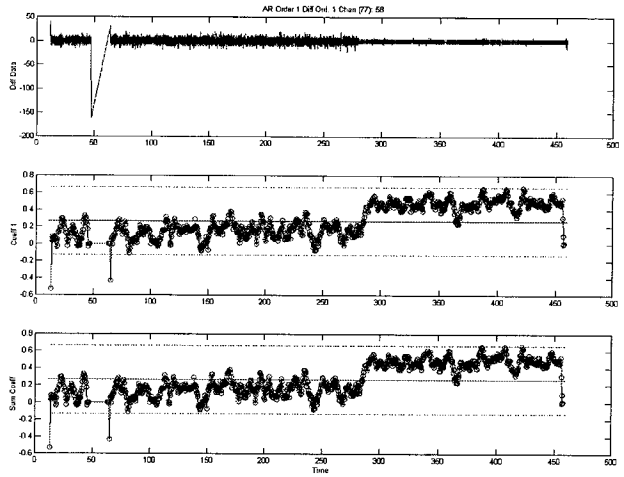

Figure 13. Mean and the 99% confidence level for the AR parameters during a frozen sense line anomaly at t = 275 sec.

In the seven anomalous data files, the DIAD was able to detect anomalies in each of the files. DIAD was able to detect all of the major anomalies, including HPFTP blade failure, HPFTP and LPFTP cavitation, HPOT performance shift, fuel flowmeter shift, frozen sense lines, and deactivated sensors. There were a couple of exceptions. DIAD was not able to identify, namely, the anomaly in the HPFTP during fuel turbine pump cavitation event. It was, however, able to detect the cavitation in the LPFTP. In addition, DIAD was able to detect some of the minor efficiency shifts in the HPOT. The cause of these deficiencies was the large variation in the training data. Overall, the DIAD was able to detect all of the major anomalies in the seven anomalous data.

7. Conclusion

The DIAD module of BEAM was used to analyze SSME ground test data. A customized GUI version of BEAM was developed for the purpose of evaluating SSME ground test data. DIAD detects anomalies by computing and examining the coefficients of an auto-regressive model. DIAD was trained using sixteen nominal test firing data from Block II engine series. It was then used to detect anomalies in seven different test data that contained some of the most commonly encountered anomalies in SSME testing.

In the seven anomalous data files, DIAD was able to detect anomalies in each file. DIAD detected all the major anomalies including HPFTP blade failure, LPFTP and HPFTP cavitation, HPOT performance shift, fuel flowmeter shift, frozen sense lines, and deactivated sensors. There were a couple of exceptions. DIAD was not able to identify the anomaly in the HPFTP during fuel turbine pump cavitation event. It was, however, able to detect the cavitation in the LPFTP. DIAD was also able to detect

some of the minor efficiency shifts in the HPOT. The cause of these deficiencies was found to be the large variations in the sixteen training data files. The large variability in some sensor data between different engine tests forced the anomaly threshold level to very high values, resulting in an insensitive detector. Overall, the DIAD was sensitive to all of the major anomalies in the seven anomalous data and detected the shift in the data characteristics. However, a few of the anomalies were missed due to the high anomaly threshold from large variability in the sensor data during training.

The future improvement to DIAD will focus on the problem of variability in the sensor data among engines. One solution is to classify nominal behavior clustering of nominal parameter values. In other words, certain groups of engines appear to exhibit similar dynamical behavior. This is reasonable since Block II engines are built in small production runs. This will require a clever clustering algorithm, which we are in the process of developing and evaluating.

REFERENCES

[1] Mackey, R., James, M., Park, H. G., Zak, M., "BEAM: Technology for Autonomous Self-Analysis," IEEE Aerospace Conference, Big Sky, Montana, March 2001.

[2] Zak, M., Park, H. G., "Gray-box Approach to Fault Diagnosis of Dynamical Systems," IEEE Aerospace Conference, Big Sky, Montana, March 2001.

[3] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, *Time Series Analysis*, Upper Saddle River, New Jersey, Prentice Hall, 1994.

Han Park *is a senior member of the technical staff in the Ultracomputing Technologies Research Group in the Information and Computing Technologies Research Section at the Jet Propulsion Laboratory, California Institute of Technology. He joined JPL in 1999 and performs research and development of fault detection/diagnosis algorithms for aircraft and spacecraft systems. He received his B.S. in M.E. at the University of California at Berkeley, S.M. in M.E. at MIT, and Ph.D. in Aeronautics at the California Institute of Technology. His research interests are in the areas of vehicle health monitoring, signal processing, image processing, pattern recognition, color recognition, quantitative visualization, fluid mechanics, and heat transfer.*

Ryan Mackey *received his B.A. degree from the University of California at Santa Cruz (1993) for Mathematics and Physics, and went on to an M.S. (1994) and Eng. (1997) in Aeronautics at Caltech. He is presently a senior researcher and charter member of the Ultracomputing Technologies Research Group at the Jet Propulsion Laboratory. His research centers upon revolutionary computing methods and technologies for advanced machine autonomy, specifically deep space missions, UAVs and maintainable aerospace vehicles. His interests also include quantum- and biologically-inspired computing.*

Mark L. James *is a senior researcher of the Ultracomputing Technologies Research Group at Jet Propulsion Laboratory. At JPL he is Principal Investigator in real-time inference and knowledge-based systems. His primary research focus is on high-speed inference systems and their application to planetary and deep spacecraft systems. His expertise includes core artificial intelligence technology, high-speed real-time inference systems and flight system software architectures. He has received a number of NASA awards that include NASA Software of the Year Award Nominee.*

Michail Zak *is a senior research scientist in the Ultracomputing Technologies Research Group at the Jet Propulsion Laboratory, California Institute of Technology. He has been with JPL since 1977. His research interests include non-linear dynamical system theory, chaos theory, quantum information processing, neural networks, and complex systems theory. He is the author of over 150 scientific and technical papers, and research monographs.*

Michael H. Kynard *is an engine systems engineer within the SSME project office at NASA's Marshall Space Flight Center. He joined NASA/MSFC in 1985 and has held positions as software engineer for the SSME Main Engine Controller, lead engineer for the SSME resident office at Stennis Space Center, and lead SSME systems engineer for the Propulsion Laboratory at MSFC. He received his B.S. in Electrical Engineering from The University of Alabama.*

John M. Sebghati *is a Subtask Lead for SSME Systems on the Vehicle and Systems Development Team of the MSFC Group of Sverdrup Technology, Inc. He joined Sverdrup in 1999 and is responsible for providing SSME test and launch support. He has also provided support for the testing of the Linear Aerospike Engine (XRS-2200). He received his B.S. in Aerospace Engineering at Iowa State University and an M.S. in Aerospace Engineering at the University of Tennessee.*

William D. Greene *is a Team Lead within the Engine Systems Performance Group of the Space Transportation Directorate, NASA Marshall Space Flight Center. He joined NASA in 2000 after spending ten years with aerospace and defense contractors working on a diverse array of programs including cryogenic propellant densification, launch vehicle main propulsion systems, liquid propellant large caliber gun internal ballistics, and liquid and hybrid rocket engine performance. He received his B.S. and M.S. in Aerospace Engineering from the Pennsylvania State University.*